Aula 04

Aritmética de ponteiros, ordenação de vetor, arrays multidimensionais

pilhas e filas

Academy

🤓 - Quem somos

Leonardo Mesquita

Luca Gabriel

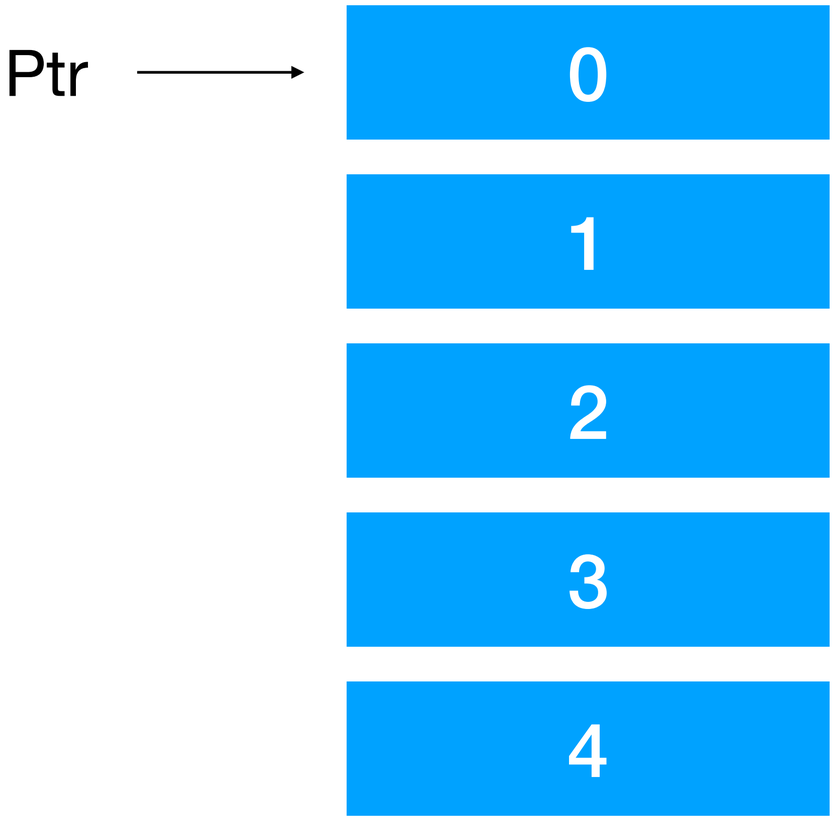
Kauã Miguel

João Victor

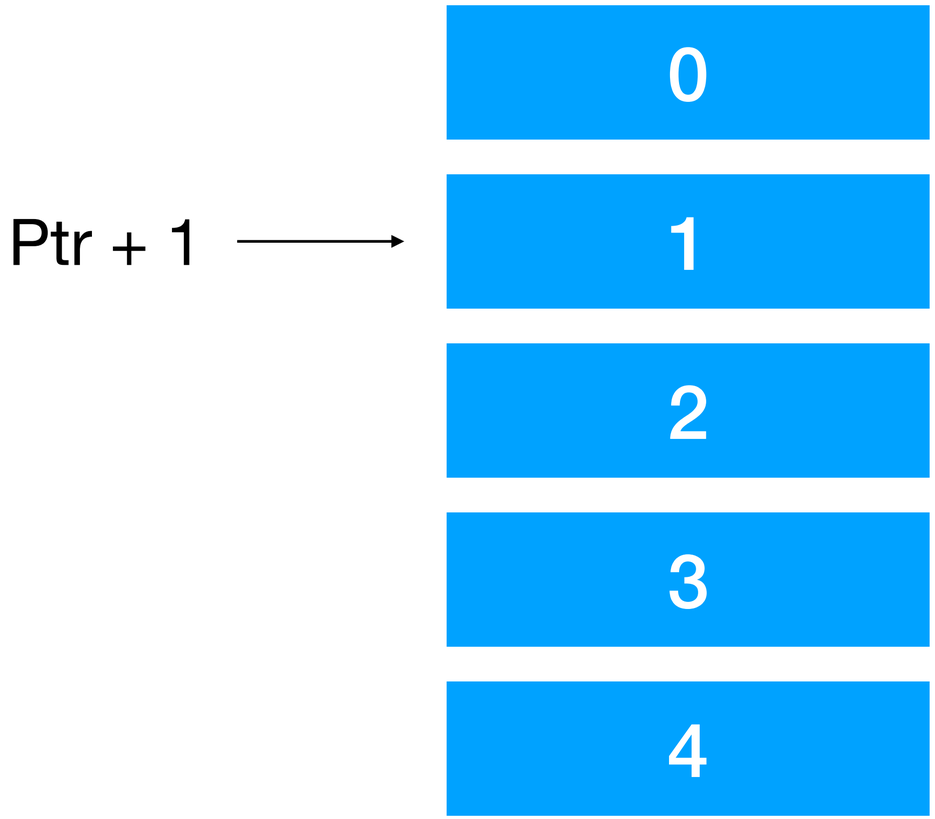
Aritmética de ponteiros

{0,1,2,3,4,5}

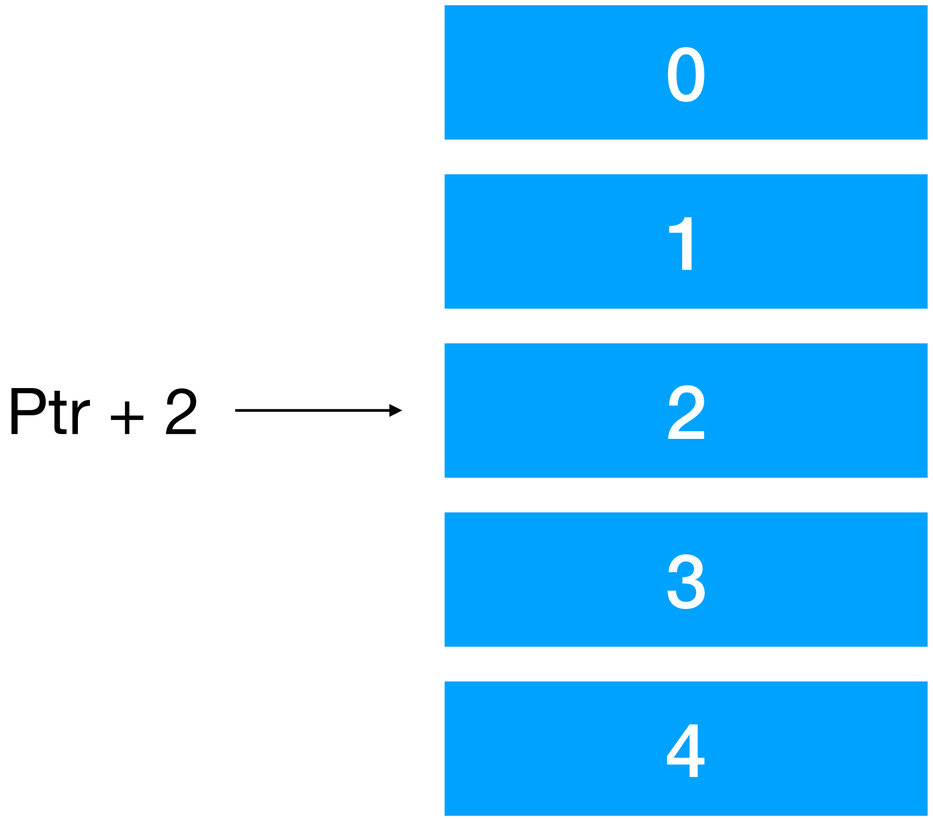
Ponteiro para o array



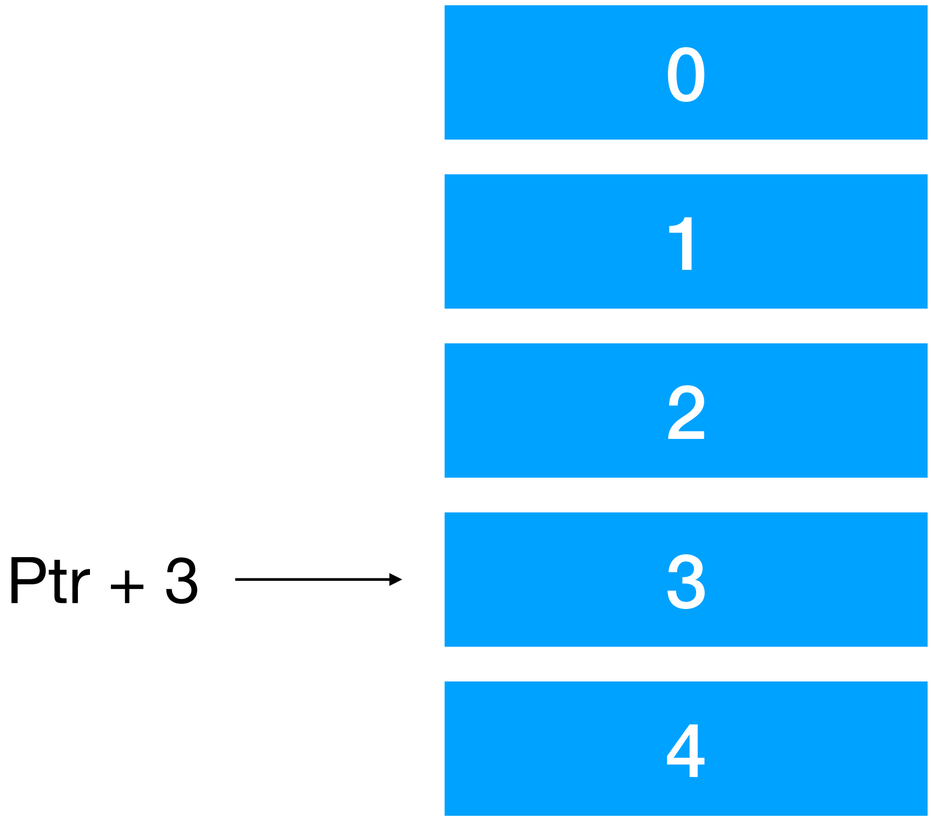
{0,1,2,3,4,5}



{0,1,2,3,4,5}



{0,1,2,3,4,5}



```c
int numeros[] = {0, 1, 2, 3, 4, 5};
int *ptr = numeros;

printf("%d", *ptr);

return 0;
```

0

```c
int numeros[] = {0, 1, 2, 3, 4, 5};
int *ptr = numeros;

for(int i = 0; i < 6; i++) {
    printf("%d", *(ptr + i));
}

return 0;
```

```c
int numeros[] = {0, 1, 2, 3, 4, 5};
int *ptr = numeros;

for(int i = 0; i < 6; i++) {
    printf("%d", *(ptr + i));
}

return 0;
```

0 1 2 3 4 5

```c
int numeros[] = {0, 1, 2, 3, 4, 5};
int *ptr = numeros;

for(int i = 0; i < 6; i++) {
    printf("%d", (*ptr + i));
}

return 0;
```

0 1 2 3 4 5

```c
int numeros[] = {5, 10, 15, 20};
int *ptr = numeros + 2;
int *ptr2 = ptr - 1;

printf("%d\n", *(ptr2 + 2));

return 0;
```

A) 5

B) 10

C) 15

D) 20

```c
int numeros[] = {5, 10, 15, 20};
int *ptr = numeros + 2;
int *ptr2 = ptr - 1;

printf("%d\n", *(ptr2 + 2));

return 0;
```

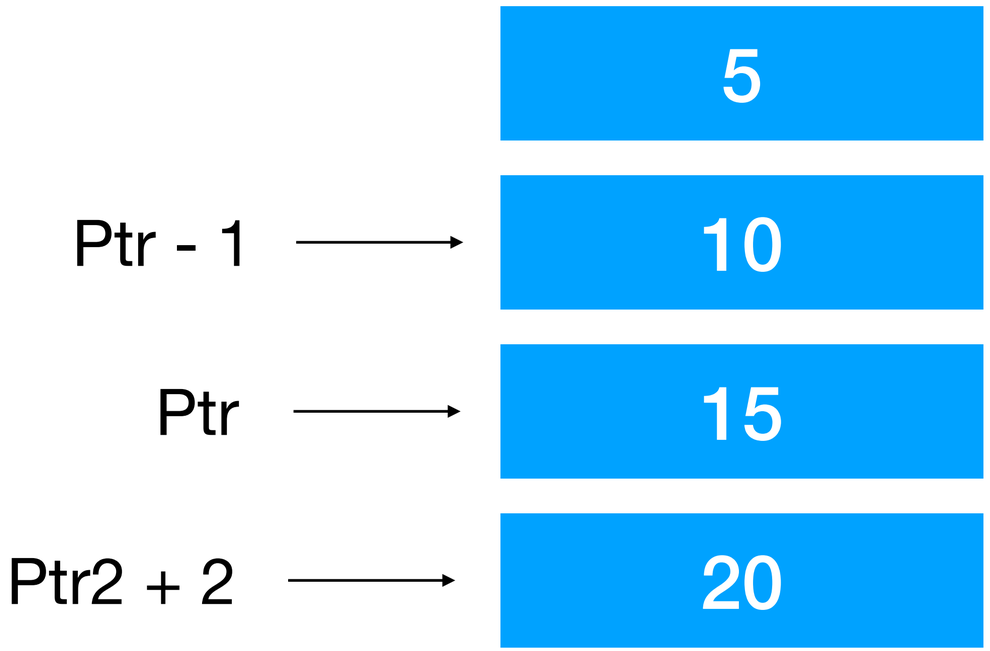
A) 5

B) 10

C) 15

D) 20

{5,10,15,20}



```
int arr[] = {5, 15, 25, 35, 45, 55, 65, 75, 85, 95};
int *p = arr;
int total = 0;

for(int i = 0; i < 10; i++) {
    if(*p % 2 == 0) {
        total += *p;
    } else {
        total += *p * 2;
    }
    p++;
}

p = arr;
while(*p != 85) {
    if(*p > 50) {
        *p = *p - 10;
    }
    p++;
}
return 0;
```

total = ?

arr = ?

```c
int arr[] = {5, 15, 25, 35, 45, 55, 65, 75, 85, 95};
int *p = arr;
int total = 0;

for(int i = 0; i < 10; i++) {
    if(*p % 2 == 0) {
        total += *p;
    } else {
        total += *p * 2;
    }
    p++;
}

p = arr;
while(*p != 85) {
    if(*p > 50) {
        *p = *p - 10;
    }
    p++;
}
return 0;
```

total = ?

arr = ?

```
int arr[] = {5, 15, 25, 35, 45, 55, 65, 75, 85, 95};
int *p = arr;
int total = 0;

for(int i = 0; i < 10; i++) {
    if(*p % 2 == 0) {
        total += *p;
    } else {
        total += *p * 2;
    }
    p++;
}

p = arr;
while(*p != 85) {
    if(*p > 50) {
        *p = *p - 10;
    }
    p++;
}
return 0;
```

total = 1000

arr = 5, 15, 25, 35, 45, 45, 55, 65, 85, 95

```c
int arr[] = {5, 15, 25, 35, 45, 55, 65, 75, 85, 95};
int *p = arr;
int total = 0;

for(int i = 0; i < 10; i++) {
    if(*p % 2 == 0) {
        total += *p;
    } else {
        total += *p * 2;
    }
    p++;
}

p = arr;
while(*p != 85) {
    if(*p > 50) {
        *p = *p - 10;
    }
    p++;
}
return 0;
```

total else 10
total else 40
total else 90
total else 160
total else 250
total else 360
total else 490
total else 640
total else 810
total else 1000

Ordenação de vetores

50 , 20 , 70 , 45 , 32 , 28

Crescente

50 , 20 , 70 , 45 , 32 , 28

Decrescente

50 , 20 , 70 , 45 , 32 , 28

Algoritmos de ordenação

Algoritmos de ordenação

Bubble sort

Heap sort

Selection sort

Merge sort

Insertion sort

Quick sort

Bubble sort

```
int arr[] = {45,23,35,76,32,40};
int i,j,temp;

for (i = 0; i < 6 - 1; i++) {
    for (j = 0; j < 6 - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
}

return 0;
```

{23, 45, 35, 76, 32, 40}

{23, 35, 45, 76, 32, 40}

{23, 35, 45, 32, 76, 40}

{23, 35, 45, 32, 40, 76}

{23, 35, 32, 45, 40, 76}

{23, 35, 32, 40, 45, 76}

{23, 32, 35, 40, 45, 76}

Quick sort

```
int arr[] = {64, 25, 12, 22};
int n = 5;
int i, j, min_idx;

for (i = 0; i < n - 1; i++) {
    min_idx = i;
    for (j = i + 1; j < n; j++) {
        if (arr[j] < arr[min_idx]) {
            min_idx = j;
        }
    }

    if (min_idx != i) {
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

return 0;
```

{12, 25, 64, 22}

{12, 22, 64, 25}

{12, 22, 25, 64}

```c
int arr[] = {64, 25, 12, 22, 30, 45};
int n = 5;

int *ptr1, *ptr2, temp;

for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        ptr1 = arr + j;
        ptr2 = arr + j + 1;

        if (*ptr1 < *ptr2) {
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
       } else {
            temp = *ptr2;
            *ptr2 = *ptr1;
            *ptr1 = temp;
       }
    }
}
```

Quantas vezes o array foi modificado?

Como o array vai ficar ao final da execução?

```c
int arr[] = {64, 25, 12, 22, 30, 45};
int n = 5;

int *ptr1, *ptr2, temp;

for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        ptr1 = arr + j;
        ptr2 = arr + j + 1;

        if (*ptr1 < *ptr2) {
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
       } else {
            temp = *ptr2;
            *ptr2 = *ptr1;
            *ptr1 = temp;
       }
    }
}
```

Quantas vezes o array foi modificado?

10

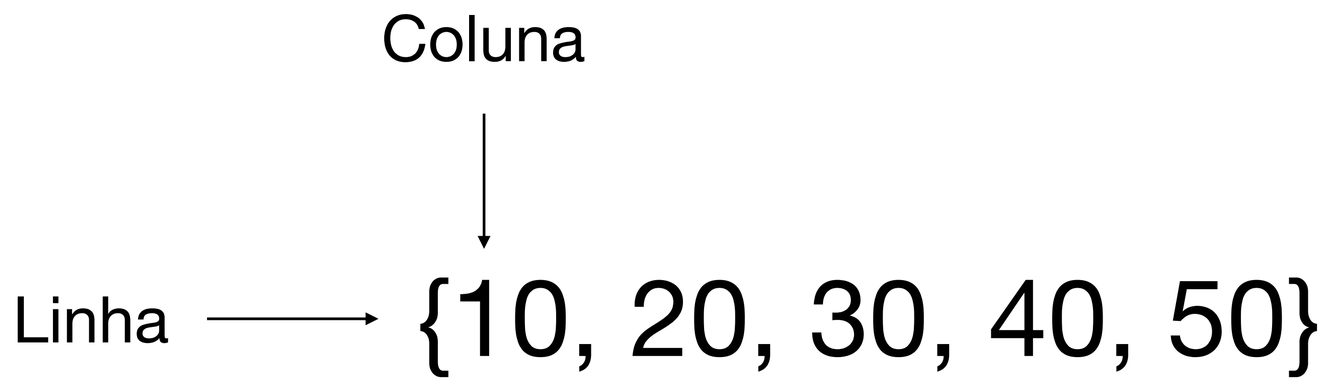
Como o array vai ficar ao final da execução?

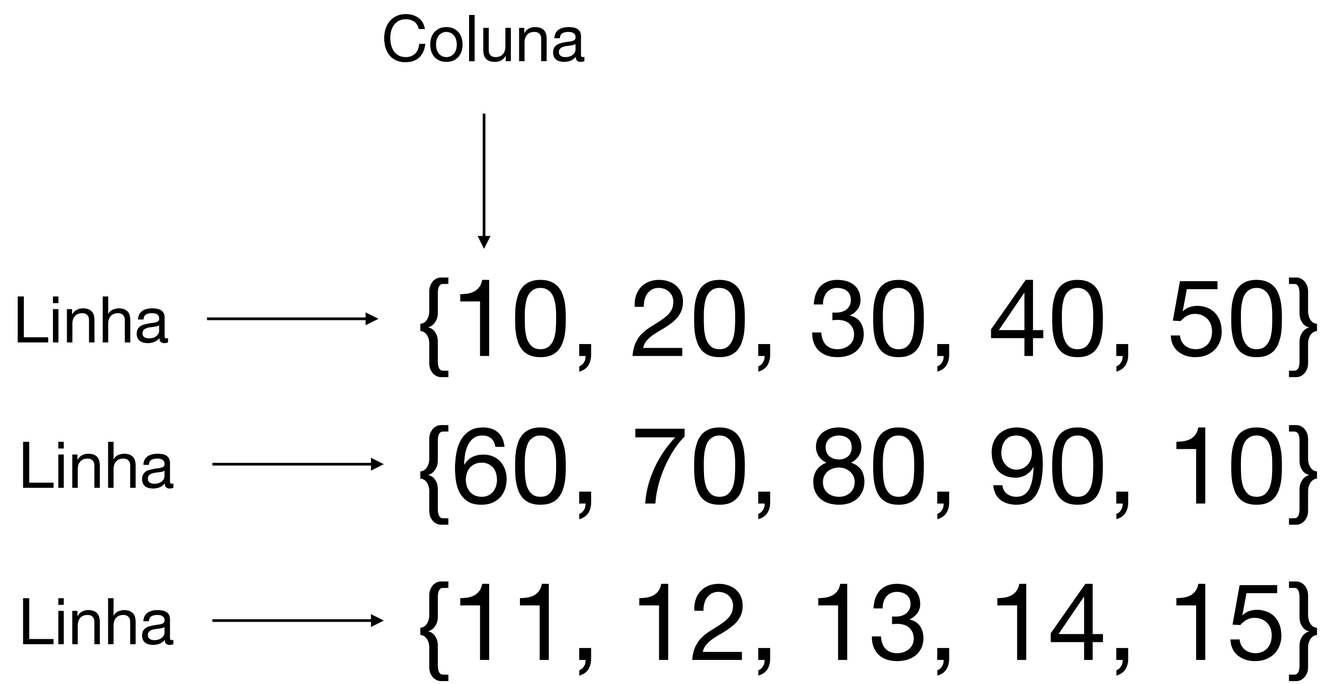
30 22 12 25 64 45

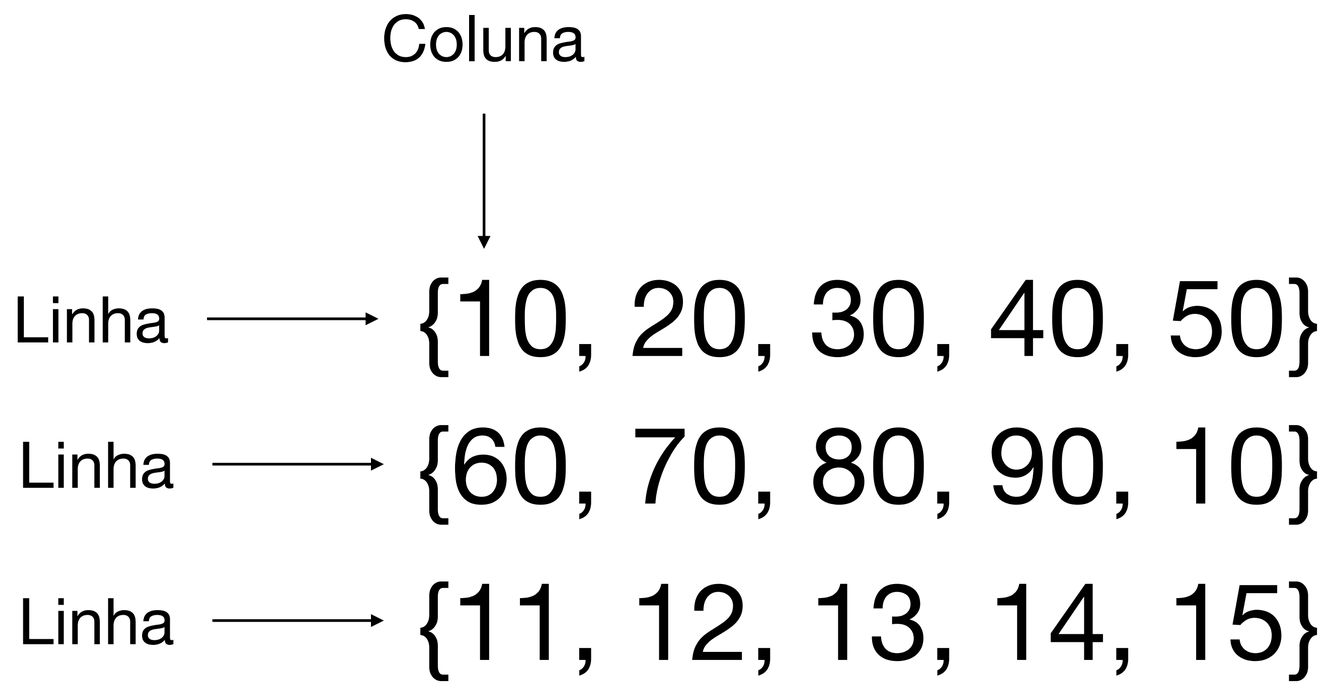
```
int arr[] = {64, 25, 12, 22, 30, 45};
int n = 5;

int *ptr1, *ptr2, temp;

for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        ptr1 = arr + j;
        ptr2 = arr + j + 1;

        if (*ptr1 < *ptr2) {
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
       } else {
            temp = *ptr2;
            *ptr2 = *ptr1;
            *ptr1 = temp;
       }
    }
}
```

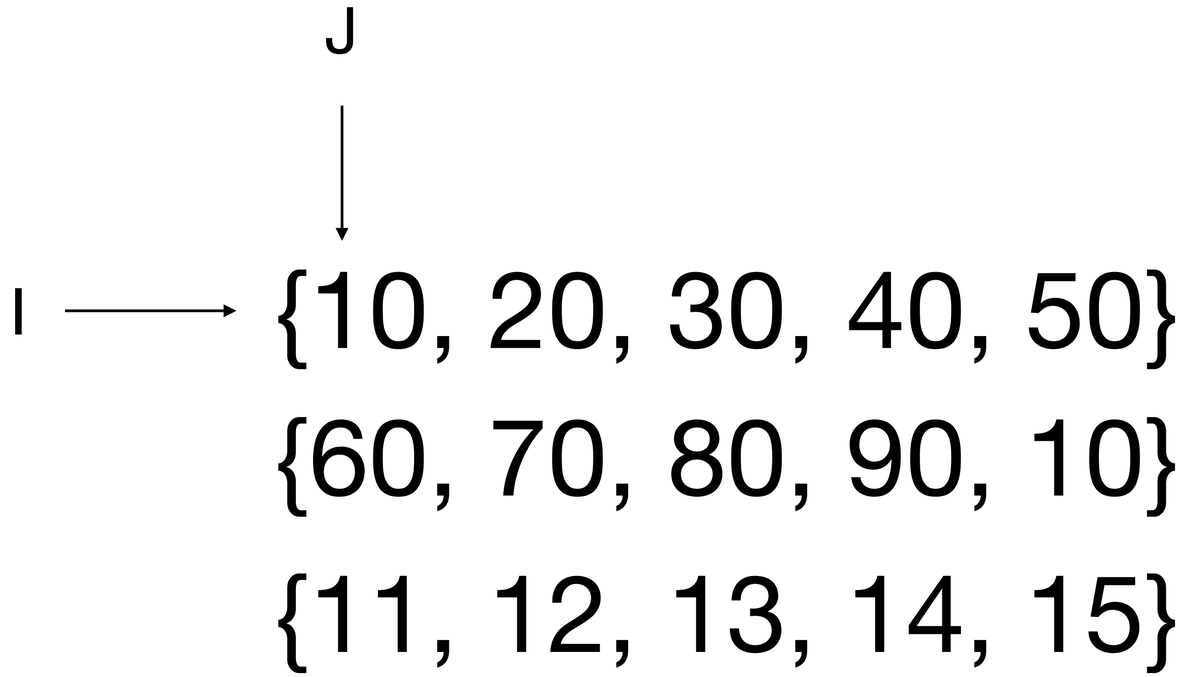
```
int arr[] = {64, 25, 12, 22, 30, 45};
int n = 5;

int *ptr1, *ptr2, temp;

for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        ptr1 = arr + j;
        ptr2 = arr + j + 1;

        if (*ptr1 < *ptr2) {
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
        } else {
            temp = *ptr2;
            *ptr2 = *ptr1;
            *ptr1 = temp;
        }
    }
}
```

Arrays Multidimensionais


Coluna
Linha
{10, 20, 30, 40, 50}


Coluna
Linha
{10, 20, 30, 40, 50}
Linha
{60, 70, 80, 90, 10}
Linha
{11, 12, 13, 14, 15}


Coluna
Linha
{10, 20, 30, 40, 50}
Linha
{60, 70, 80, 90, 10}
Linha
{11, 12, 13, 14, 15}


J
I
{10, 20, 30, 40, 50}
{60, 70, 80, 90, 10}
{11, 12, 13, 14, 15}

{10, 20, 30, 40, 50}

{60, 70, 80, 90, 10}

{11, 12, 13, 14, 15}

{10, 20, 30, 40, 50}

{60, 70, 80, 90, 10}

{11, 12, 13, 14, 15}

J = 1

{10, 20, 30, 40, 50}

I = 1 {60, 70, 80, 90, 10}

{11, 12, 13, 14, 15}

A11

J = 1

{10, 20, 30, 40, 50}

I =1 {60, 70, 80, 90, 10}

{11, 12, 13, 14, 15}

```
int matrix[ROWS][COLS] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
 };
```

```
int matrix[ROWS][COLS] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
 };
```

```
int matrix[ROWS][COLS] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
 };
```

matrix[1][2]

```
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", matrix[i][j]);
    }
    printf("\n");
}
```

```
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", matrix[i][j]);
    }
    printf("\n");
}
```

```
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", matrix[i][j]);
    }
    printf("\n");
}
```

```
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", matrix[i][j]);
    }
    printf("\n");
}
```

```c
#define ROWS_A 3
#define COLS_A 2
#define ROWS_B 2
#define COLS_B 4

int main(void) {
    int first[ROWS_A][COLS_A] = {
        {1, 2},
        {3, 4},
        {5, 6}
    };

    int second[ROWS_B][COLS_B] = {
        {7, 8, 9, 10},
        {11, 12, 13, 14}
    };

    int result[ROWS_A][COLS_B];

    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            result[i][j] = 0;
            for (int k = 0; k < COLS_A; k++) {
                result[i][j] += first[i][k] * second[k][j];
            }
        }
    }
```

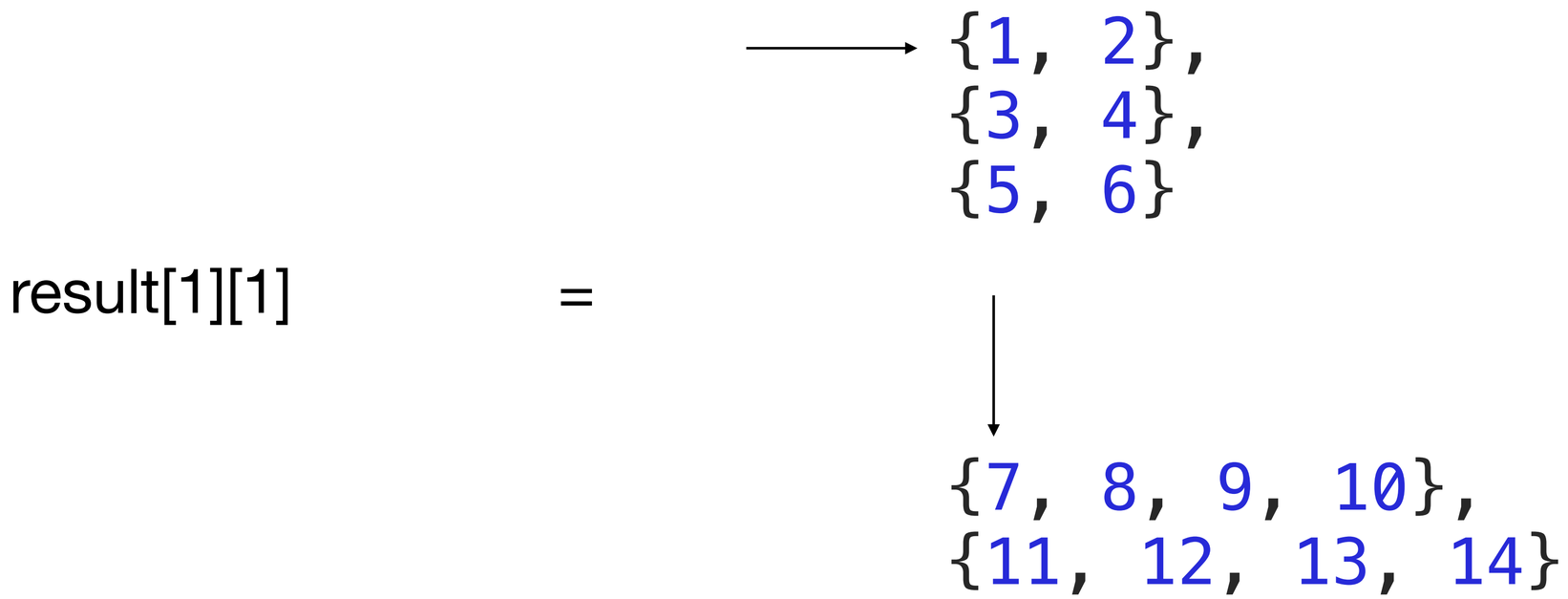
Qual será o valor dos elementos da matriz result?

```c
#define ROWS_A 3
#define COLS_A 2
#define ROWS_B 2
#define COLS_B 4

int main(void) {
    int first[ROWS_A][COLS_A] = {
        {1, 2},
        {3, 4},
        {5, 6}
    };

    int second[ROWS_B][COLS_B] = {
        {7, 8, 9, 10},
        {11, 12, 13, 14}
    };

    int result[ROWS_A][COLS_B];

    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            result[i][j] = 0;
            for (int k = 0; k < COLS_A; k++) {
                result[i][j] += first[i][k] * second[k][j];
            }
        }
    }
```

Qual será o valor dos elementos da matriz result?

29 32 35 38
65 72 79 86
101 112 123 134

Multiplicação de matrizes



Multiplicação de matrizes

result[1][1] = (1 * 7) + (11 * 2)

Multiplicação de matrizes

result[1][1] = (1 * 7) + (11 * 2)

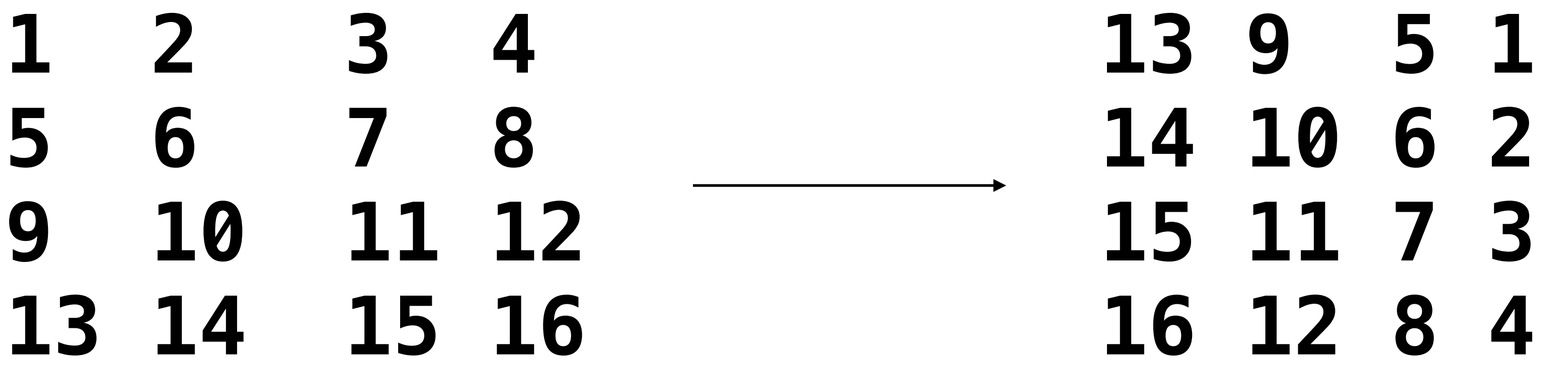
```java
int n = 4;
int matrix[n][n];

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matrix[i][j] = i * n + j + 1;
    }
}

int temp[n][n];

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        temp[j][n - 1 - i] = matrix[i][j];
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matrix[i][j] = temp[i][j];
    }
}
```

Qual será o valor dos elementos da matriz?

```java
int n = 4;
int matrix[n][n];

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matrix[i][j] = i * n + j + 1;
    }
}

int temp[n][n];

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        temp[j][n - 1 - i] = matrix[i][j];
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matrix[i][j] = temp[i][j];
    }
}
```

Qual será o valor dos elementos da matriz?

13 9 5 1
14 10 6 2
15 11 7 3
16 12 8 4


1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
13 9 5 1
14 10 6 2
15 11 7 3
16 12 8 4

Pilhas e filas

Pilhas/Stack

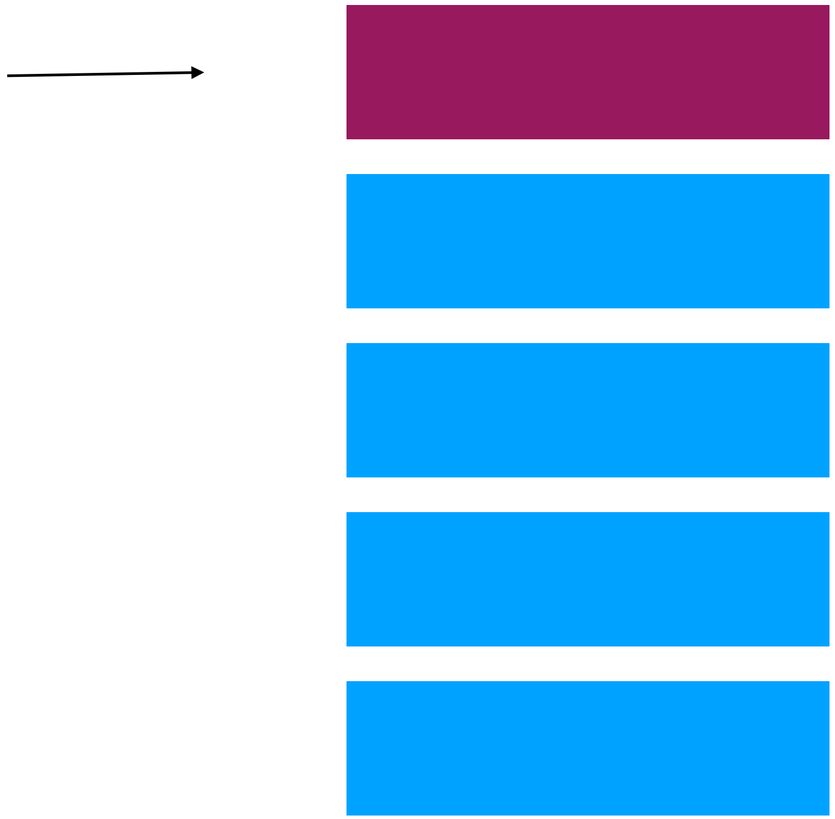
Pilhas/Stack

LIFO - Last In First Out

Pilhas/Stack

LIFO - Last In First Out

Pilhas/Stack



Pilhas/Stack

Pop

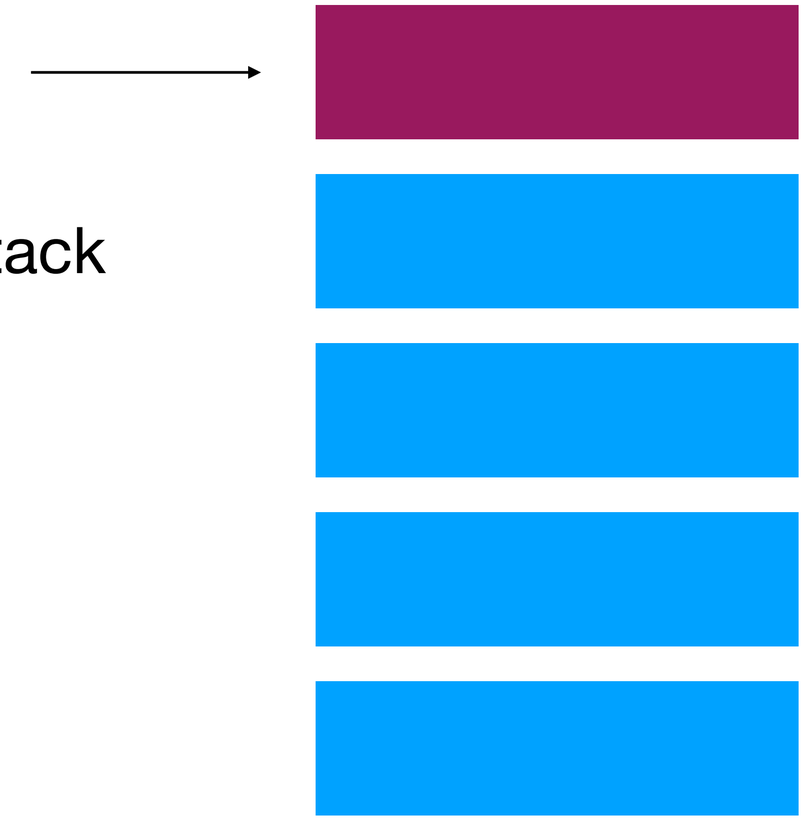
Push

Pilhas/Stack

Pop - retirar um elemento da stack

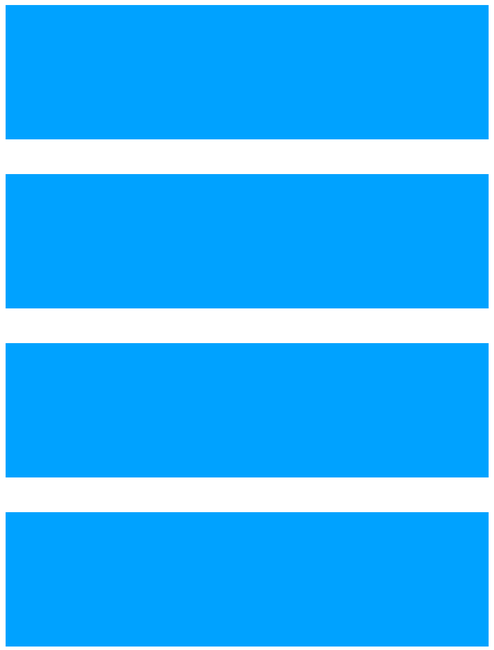
Pilhas/Stack

Pop - retirar um elemento da stack



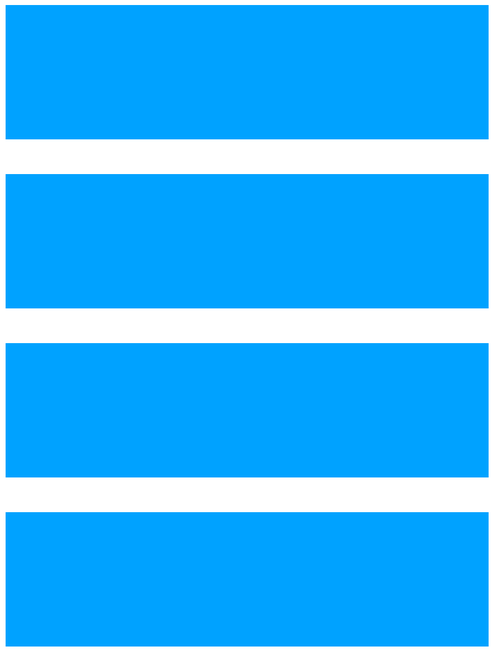
Pilhas/Stack

Pop - retirar um elemento da stack



Pilhas/Stack

Push - Adicionar elementos



Pilhas/Stack

Push - Adicionar elementos

Pilhas/Stack

Push - Adicionar elementos

Sempre adicionar ao topo da stack

Pilhas/Stack

```
struct Stack {
    int items[MAX];
    int top;
};
```

Pilhas/Stack

```
struct Stack {
    int items[50];
    int top;
};
void initStack(struct Stack* s) {
    s->top = -1;
}
int isFull(struct Stack* s) {
    return s->top == 50 - 1;
}

int isEmpty(struct Stack* s) {
    return s->top == -1;
}
```

Pilhas/Stack

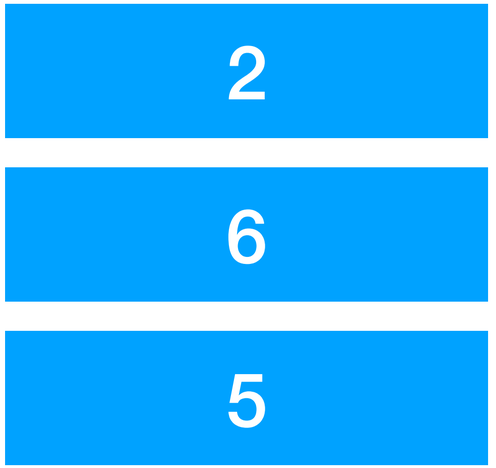
```
struct Stack {
    int items[50];
    int top;
};

void push(struct Stack* s, int value) {
    if (isFull(s)) {
        printf("A pilha está cheia.");
    } else {
        s->top++;
        s->items[s->top] = value;
    }
}
```

Pilhas/Stack

```
 struct Stack {
     int items[50];
     int top;
 };

int pop(struct Stack* s) {
    if (isEmpty(s)) {
        return -1;
    } else {
        int poppedItem = s->items[s->top];
        s->top--;
        return poppedItem;
    }
}
```

Pilhas/Stack

```
struct Stack stack;

initStack(&stack);

push(&stack, 5);
push(&stack, 6);
push(&stack, 2);

pop(&stack);
pop(&stack);
```



Fila/Queue

Fila/Queue

FIFO - First In First Out

Fila/Queue

FIFO - First In First Out

Fila/Queue

FIFO - First In First Out

Fila/Queue

Enqueue

Dequeue

Fila/Queue

Enqueue - Adicionar um elemento ao final da fila

Fila/Queue

Enqueue - Adicionar um elemento ao final da fila

Fila/Queue

Dequeue - Remover o primeiro elemento da fila

Fila/Queue

Dequeue - Remover o primeiro elemento da fila

Fila/Queue

```
struct Queue {
    int items[50];
    int front;
    int rear;
};
```

Fila/Queue

```
struct Queue {
    int items[50];
    int front;
    int rear;
};
void initQueue(struct Queue* q) {
    q->front = -1;
    q->rear = -1;
}

int isFull(struct Queue* q) {
    return q->rear == 50 - 1;
}

int isEmpty(struct Queue* q) {
    return q->front == -1 || q->front > q->rear;
}
```

Fila/Queue

```
struct Queue {
    int items[50];
    int front;
    int rear;
};

void enqueue(struct Queue* q, int value) {
    if (isFull(q)) {
        printf("A fila está cheia");
    } else {
        if (q->front == -1) {
            q->front = 0;
        }
        q->rear++;
        q->items[q->rear] = value;
    }
}
```

Fila/Queue

```
struct Queue {
    int items[50];
    int front;
    int rear;
};


int dequeue(struct Queue* q) {
    if (isEmpty(q)) {
        return -1;
    } else {
        int dequeuedItem = q->items[q->front];
        q->front++;
        return dequeuedItem;
    }
}
```

Fila/Queue

```
struct Queue queue;

initQueue(&queue);

enqueue(&queue, 10);
enqueue(&queue, 25);
enqueue(&queue, 90);

dequeue(&queue);
```

Fila/Queue

```
struct Queue queue;

initQueue(&queue);

enqueue(&queue, 10);
enqueue(&queue, 25);
enqueue(&queue, 90);

dequeue(&queue);
```

Fila/Queue

```
struct Queue queue;

initQueue(&queue);

enqueue(&queue, 10);
enqueue(&queue, 25);
enqueue(&queue, 90);

dequeue(&queue);
```

Fila/Queue

```
struct Queue queue;

initQueue(&queue);

enqueue(&queue, 10);
enqueue(&queue, 25);
enqueue(&queue, 90);

dequeue(&queue);
```

Fila/Queue

```
struct Queue queue;

initQueue(&queue);

enqueue(&queue, 10);
enqueue(&queue, 25);
enqueue(&queue, 90);

dequeue(&queue);
```

Questão

```c
int N = 3, M = 3;
int matriz[3][3] = {
    {1, 0, 3},
    {0, 5, 6},
    {7, 8, 9}
};
int matriz2[3][3];
int* ptrMatriz = &matriz[0][0];
int* ptrMatriz2 = &matriz2[0][0];

for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        *(ptrMatriz2 + j * N + i) = *(ptrMatriz + i * M + j);
    }
}
```

Qual serão os valores dos índices da matriz2?

```c
int N = 3, M = 3;
int matriz[3][3] = {
    {1, 0, 3},
    {0, 5, 6},
    {7, 8, 9}
};
int matriz2[3][3];
int* ptrMatriz = &matriz[0][0];
int* ptrMatriz2 = &matriz2[0][0];

for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        *(ptrMatriz2 + j * N + i) = *(ptrMatriz + i * M + j);
    }
}
```

Qual serão os valores dos índices da matriz2?

1 0 7
0 5 8
3 6 9

Linkedin

Luca Lacerda